

043290.P6156

Patent

UNITED STATES PATENT APPLICATION

for

*Code Sequence*  
~~METHOD AND APPARATUS FOR~~  
VECTOR GATHER AND SCATTER

INVENTORS:

**Carole Dulong**

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP  
12400 WILSHIRE BOULEVARD  
SEVENTH FLOOR  
LOS ANGELES, CALIFORNIA 90025  
(408) 720-8300

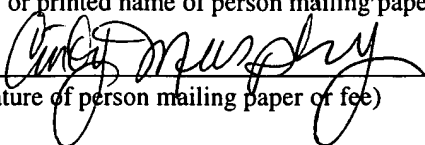
Attorney's Docket No. 042390.P6156

"Express Mail" mailing label number EL431886484 US

Date of Deposit March 29, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Cindy Murphy  
(typed or printed name of person mailing paper or fee)

  
(Signature of person mailing paper or fee)

006220 21082560

## BACKGROUND OF THE INVENTION

Field of the Invention

5 This invention relates generally to the field of processor operations. More particularly, the invention relates to an apparatus and method for performing vector gather and scatter operations using a computer processor.

Description of the Related Art

10 In order to perform vector computations on a computer, matrices such as that illustrated in Figure 1 must frequently be loaded into memory. Once in memory, the matrix may be combined with other matrices (not shown) to perform complex, multidimensional computations (e.g., vector addition, vector multiplication).

15 One problem which exists, however, is that matrices can take up a substantial amount of memory, particularly when used to store certain types of data (e.g., scientific data pertaining to physical phenomenon). In addition, matrices may be sparsely populated with data elements. For example, only 4 data elements out of the 24 illustrated in Figure 1 contain non-zero values, resulting in an inefficient use of memory.

20 To conserve memory when working such large, sparsely populated matrices, "gather" and "scatter" operations were developed. For example, the

CRAY-1 computer system performed gather operations to collect the elements of a matrix from memory and store them in a highly compressed format (e.g., sorted contiguously in an ordered array). Conversely, when necessary to perform various matrix operations (e.g., matrix multiplication) the CRAY-1

5 performed scatter operations to reproduce the previously-gathered matrix in memory.

One problem which exists, however, is that these systems require complex dedicated hardware to perform the gather and scatter operations. For example, the CRAY-1 employed a vector processor which performed gather and scatter

10 operations using dedicated registers to hold index vectors and dedicated address calculation hardware.

Accordingly, what is needed is a more efficient apparatus and method for storing and working with matrices in a computing environment. What is also needed is a system and method for performing gather and scatter operations on

15 a general purpose processor.

## BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

5        **FIG. 1** illustrates matrix with data elements which may be stored in a computer memory.

**FIG. 2** illustrates an exemplary computer architecture used to implement elements of the invention.

10        **FIG. 3** illustrates a variety of data and data storage formats according to embodiments of the invention.

**FIG. 4** illustrates extract and deposit operations according to embodiments of the invention.

**FIG. 5** illustrates one embodiment of a method for performing a gather operation.

15        **FIG. 6** illustrates the extraction of a set of address indices according to one embodiment of the invention.

**FIG. 7** illustrates address calculation and storage operations according to one embodiment of the invention.

FIG. 8 illustrates memory load operations according to one embodiment of the invention.

FIG. 9 illustrates the merging of data elements in a register according to one embodiment of the invention.

5        FIG. 10 illustrates one embodiment of a method for performing a scatter operation.

FIG. 11 illustrates performing an extract operation on a plurality of data elements according to one embodiment of the invention.

006260-2108E560

## DETAILED DESCRIPTION

In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form to avoid obscuring the underlying principles of the present invention.

Embodiments of the present invention include various steps, which will be described below. The steps may be embodied in machine-executable code.

The instructions can be used to cause a general-purpose or special-purpose processor to perform certain steps. Alternatively, these steps may be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

### AN EXEMPLARY COMPUTER SYSTEM

**Figure 2** shows a computer system 200 upon which embodiments of the invention may be implemented. Computer system 200 comprises a bus 201 for communicating information, a processor 210 coupled to the bus 201 for processing information, and a memory subsystem 204-206 coupled to bus 201 for storing information and instructions for the processor 210. The memory

subsystem may be comprised of a main memory 204, a read only memory 206 and/or a mass storage device 205.

The processor 210 includes an execution unit 230, a register file 250, a cache memory 260, a decoder 265, and an internal bus 270. The cache memory 260, storing frequently and/or recently used information for the processor 210, is coupled to the execution unit 230. Register file 250 is comprised of a group of registers for storing data to be read by the execution unit 230 via the internal bus 270. In one embodiment, the registers within the register file 250 store sixty-four bits of packed data for integer and/or floating point calculations.

The execution unit 230 operates on packed data according to the instructions received by processor 210 that are included in a packed instruction set 240. The execution unit 230 also operates on non-packed data according to instructions implemented in general-purpose processors. In one embodiment the processor 210 is an Explicitly Parallel Instruction Computing ("EPIC") processor (e.g., employing the IA-64 parallel architecture developed by Intel®), capable of executing multiple instructions per clock cycle. In addition, processor 210 in one embodiment is capable of supporting the Intel Itanium™ microprocessor instruction set as well as the packed instruction set 240. Other instruction sets, such as the Pentium®, PowerPC™ and the Alpha® processor instruction sets may also be used in accordance with the described invention. Pentium and Itanium are trademarks of Intel Corporation. PowerPC™ is a trademark of IBM,

APPLE COMPUTER, and MOTOROLA. Alpha™ is a trademark of Digital Equipment Corporation.

Still referring to Figure 2, computer system 200 can also be coupled to a second I/O bus 250 via an I/O interface 230. A plurality of I/O devices may be coupled to I/O bus 250, including, for example, a display device 243, an alphanumeric input device 242 (e.g., a keyboard), a cursor control device 241 and/or a communication device 240. The communication device 240 is for accessing other computers and may comprise a modem, a network interface card, or other well known interface device, such as those used for coupling to Ethernet, token ring, or other types of networks.

#### DATA AND STORAGE FORMATS

Figure 3 illustrates three packed data-types: packed byte 301, packed word 302, and packed doubleword (dword) 303. Packed byte 301 is sixty-four bits long containing eight packed byte data elements. Generally, a data element is an individual piece of data that is stored in a single register (or memory location) with other data elements of the same length. In packed data sequences, the number of data elements stored in a register is the register size (e.g., 64-bits in the embodiment illustrated in Figure 3) divided by the length in bits of a data element. Although the registers illustrated in Figure 3 and described throughout the specification are 64-bit registers, it should be noted that the underlying principles of the invention may be implemented on registers of virtually any size.

006250" 21085560



## EXTRACT AND DEPOSIT OPERATIONS

Figures 4a and 4b illustrate two data operations which may be used in one embodiment of the invention. As illustrated in Figure 4a, an "extract" operation involves copying a specified bit field from a source register  $R_s$  to an aligned position within a destination register  $R_d$  (i.e., the least significant bit (LSB) of the bit field is aligned with bit zero of the destination register  $R_d$ ). Conversely, a "deposit" operation, as illustrated in Figure 4b, copies a specified bit field from an aligned position in a source register  $R_s$  to a specified location within a destination register  $R_d$ .

In one embodiment, individual extract and deposit instructions are included in the packed instruction set 240. Accordingly, the extract instruction may be used to copy a data element from a source register to an aligned position in a destination register. For example, the instruction  $\text{EXTR } R_d = R_s, 32, 16$  copies a data element 16 bits in length located at bit 32 in the source register (i.e., the LSB of the data element is positioned at bit 32 of the source register) to an aligned position in a destination register as illustrated in Figure 4a.

Similarly, a deposit instruction may be used to copy a data element aligned in a source register to a specified position in a destination register. For example, the instruction  $\text{DEP } R_d = R_s, 16, 32$  copies a 16 bit data element aligned in a source register to a position starting at bit 32 (i.e., the LSB of the data element is aligned with bit 32 of the destination register as illustrated in Figure

4b). In this embodiment, the  $R_p$  designation to the right of the equal sign indicates that data elements stored in the remaining bit positions of the destination register should not be overwritten (e.g., with zeros). As described below, this feature allows a series of packed data elements to be merged into a  
5 single register.

### GATHER OPERATION

In one embodiment of the apparatus and method, extract and deposit operations are used to perform "gather" operations in which non-zero data elements of a matrix are retrieved (i.e., "gathered") from memory and stored in a  
10 contiguous manner.

As set forth in the flowchart in Figure 5, in one embodiment, a plurality of address indices are extracted into an equal plurality of destination registers (at 510). Each of the indices, when combined with a base address, specifies an address in memory where a matrix data element is stored. For example, as  
15 illustrated in Figure 6, four indices I0, I1, I2, and I3 packed in a single register, R3, are extracted into four individual registers, R5, R8, R11, and R14, respectively. Four extract instructions (e.g., EXTR R5 = R3, 0, 16 for I0) may be executed to perform this operation. In the particular embodiment illustrated in Figure 6 each of the indices are 16-bits in length. However, it should be noted  
20 that indices of varying lengths may also be used in accordance with the underlying principles of the invention.

Addresses for each of the data elements are then computed at 520 (Figure 5) by adding each of the indices to the base address stored in R2. Thus, in the embodiment illustrated in Figure 7, the base address is added to each of the indices in R5, R8, R11, and R14 and the result (i.e., the addresses in memory of each of the data elements) are stored in registers R6, R9, R12 and R15, respectively.

The processor 210, at 530 (Figure 5), then loads the data elements from memory into a group of registers. For example, in the embodiment illustrated in Figure 8, data elements E0, E1, E2, and E3 are loaded from memory (after being identified via the calculated addresses) into registers R7, R10, R13 and R16, respectively.

At 540 (Figure 5), the data elements are merged into a single register. In one embodiment, this is accomplished using deposit operations. For example, referring to Figure 9, a series of deposit operations copy, in succession, E0, E1, E2, and E3 into register R4. The end result is that data elements E0-E3, which may have been scattered throughout a matrix, are now stored contiguously in register R4 (and/or a mass storage device), thereby preserving a substantial amount of memory.

## SCATTER OPERATION

The matrix containing data elements E0-E3 may need to be reconstructed in memory from time to time so that matrix operations can be performed (e.g., matrix multiplication, addition . . . etc). In one embodiment, a "scatter" operation is used to carry out this function. Referring to Figure 10, in one embodiment of the scatter operation, indices are extracted (at 1010) and added to a base address to compute the addresses in memory to which the data elements will be scattered (at 1020). This portion of the scatter operation may be similar to the first portion of the gather operation described above (e.g., 510, 520 of Figure 5).

At 1030 the data elements are extracted from the register into which they were merged. Thus, as illustrated in Figure 11, each of the data elements E0, E1, E2 and E3 are extracted from register R4 and copied into registers R7, R10, R13, and R16, respectively (e.g., for element E2 the extract instruction might read EXTR R13 = R4, R13, 32, 16). Finally, at 1040, the data elements are stored to memory based on their previously-calculated addresses. A store instruction such as STORE [R12] = R13 may be executed by the processor 210 to perform this function (i.e., the data element from R13 is stored to the memory location found in R12).

Throughout the foregoing description, for the purposes of explanation, numerous specific details were set forth in order to provide a thorough

understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without some of these specific details. Accordingly, the scope and spirit of the invention should be judged in terms of the claims which follow.

006220" 2103E560